Installation

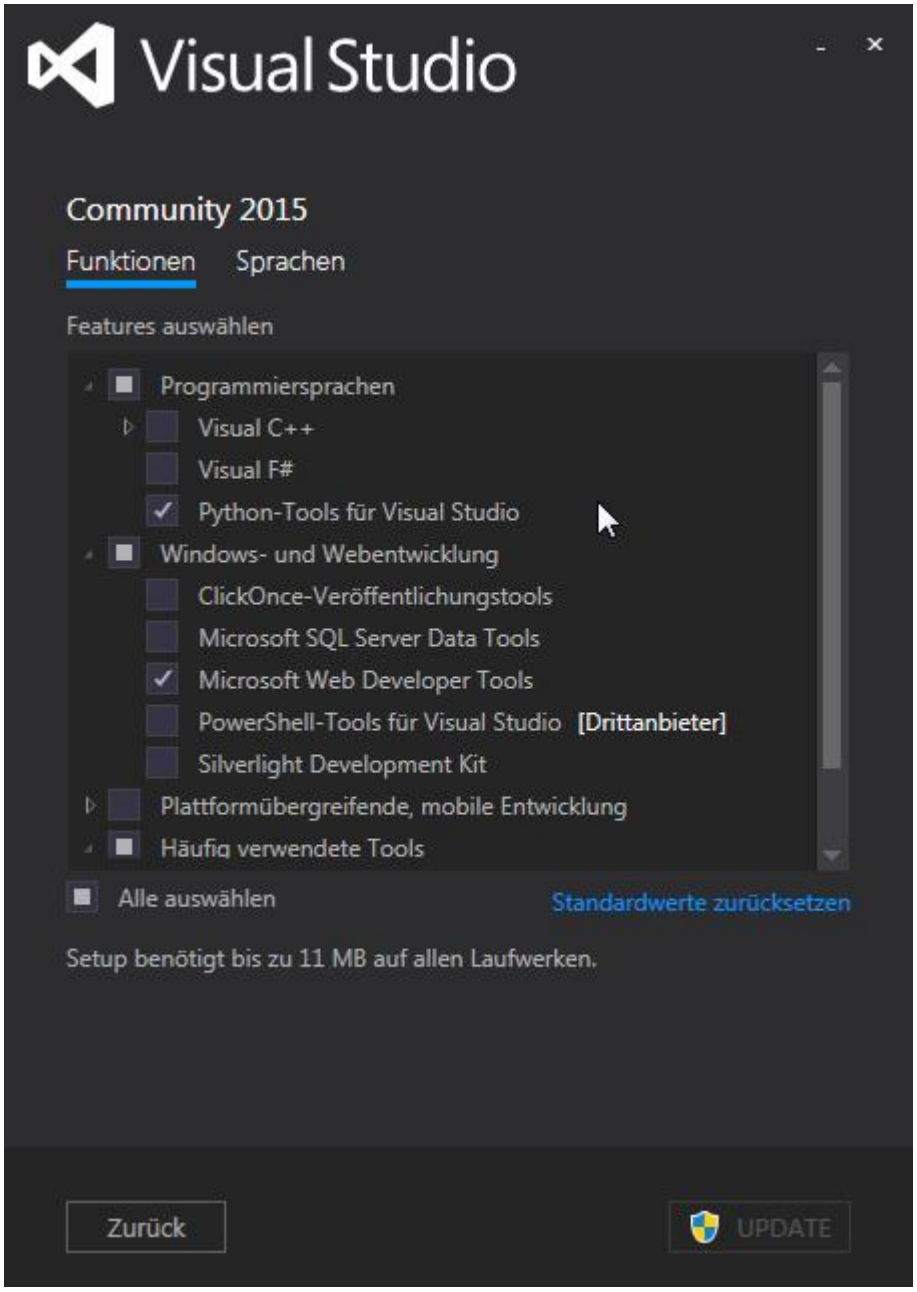
Ich versuche mal möglichst End-User-Tauglich das Setup zu beschreiben - es ist allerdings eine Menge "Holz" und erfordert sehr viele manuelle Schritte...

Disclaimer: Ihr bekommt die von mir geschriebenen Konverter so wie sie sind (as is). Es sind alle Sourcen inkludiert und ihr könnt damit machen, was immer ihr wollt. Es gibt keinerlei Garantie, dass alles problemlos funktioniert und dass es keinerlei Schäden bzw. Nebeneffekte hat. Bitte macht regelmäßig ein Backup! Durch den Generator werden immer wieder Dateien (ALLE *.conf-Files) überschrieben, da sie neu generiert werden. Wenn ihr euch irgendetwas zerschießt, so geschieht das auf eigene Verantwortung. Die von mir unten beschriebene Arbeitsweise arbeitet immer auf Kopien und bewahrt einen etwas vor trivialen Datei-Überschreibungen.

Viel Spaß!

Installation von Visual Studio:

Man braucht unbedingt Visual Studio 2015 (das ist das neueste, inzwischen auch als Final verfügbar). Es reicht die kostenlose Community Edition, die gibt es hier: https://www.visualstudio.com/downloads/download-visual-studio-vs. Bei der Installation muss bei den "Optionalen Features" unbedingt ein Haken bei "Microsoft Web Developer Tools" an sein (war aber bei mir der Default). Ferner muss bei "Programmiersprachen" ein Haken bei "Python Tools for Visual Studio" gesetzt sein (Das ist nicht Default). Weiterhin muss noch "GitHub-Integration für Visual Studio" unter "Häufig verwendete Tools" angehakt werden.



Python Tools erlauben es, auch python Files komfortabel editieren zu können (mit syntax highlighting und code completion), allerdings gehe ich in der Anleitung nicht darauf ein. Die Tools sind trotzdem nötig, da die von mir vorgeschlagene Projektstruktur ein Python-Projekt enthält.

Python Tools for Visual Studio ihrerseits brauchen noch einen Python-Interpreter, da möchte ich euch auf deren Hilfe-Seite verweisen https://github.com/Microsoft/PTVS/wiki/Selecting-and-Installing-Python-Interpreters. Ich habe bei mir CPython installiert, allerdings noch nicht viel damit gemacht.

Damit man mit dem Projekt einfach arbeiten kann und Updates auf

Knopfdruck funktionieren, habe ich alles über GitHub veröffentlicht. Da Visual Studio 2015 eine komfortable GitHub-Anbindung hat, möchte ich im folgenden das Setup auf diesem Wege beschreiben.

Damit die Integration in Visual Studio 2015 problemlos klappt, braucht mal einen GitHub-Account. Dieser ist kostenlos und kann direkt über Visual Studio 2015 eingerichtet werden.

Startet nach der Installation von Visual Studio 2015 dieses. Der erste Start dauert eine Weile. Dann gehe auf "Team->Manage Connections...". Rechts erscheint dann eine Spalte, in der neben Visual Studio Online auch GitHub steht. Dort gibt es "Connect" für diejenigen, die schon einen GitHub-User haben und "Sign Up" für neue User. Hier bitte anmelden bzw. einen neuen Account eröffnen.

Jetzt bitte folgende URL http://github.com/mumpf/smarthome-json-2-conf aufrufen, sich anmelden und auf der Seite den Button "Open in Visual Studio" klicken. Jetzt geht Visual Studio auf und in der rechten Spalte sieht man einen "Clone"-Button, den klicken.

Nun werden die Files runtergeladen und der Clone-Button wird durch einen "smarthome-json-2-conf" Eintrag ersetzt. Auf diesen ein Doppelklick. Auf fast der gleichen Stelle erscheint ein "Konverter.sln", auch hier ein Doppelklick.

Jetzt noch im Menü "Build->Rebuild Solution" ausführen, dann ist das Projekt erstmal installiert und syntaktisch korrekt. Das Projekt wurde in den Repo-Pfad von Visual Studio 2015 runtergeladen, der ist standardmäßig

"C:\Users\<username>\Source\Repos".

Das Projekt selbst liegt somit unter

"C:\Users\<username>\Source\Repos\smarthome-json-2-conf"

Setup Projektstruktur:

Mein Konverter verändert ausschließlich das items-Verzeichnis von sh.py. Optional kann man noch eine Referenz auf das *.knxproj file angeben (Projektexport aus der ETS4 bzw. ETS5), dann werden die dort definierten GA mit den in sh.py benutzten GA verglichen und Delta-Listen erzeugt. Falls ein *.knxproj-File angegeben wird, wird es ins %TEMP%/KNX entpackt und das entpackte File dort belassen (nicht wieder gelöscht), um beim nächsten konvertieren nicht wieder entpacken zu müssen, falls sich am *.knxproj nichts verändert hat. Als Voraussetzung braucht mal also Schreib- und Leserechte auf das items- und das %TEMP%-Verzeichnis.

Beim Item-Verzeichnis könnte man dieses über SAMBA anbinden und direkt

darauf arbeiten, allerdings ist das eine schlechte Idee: Die Konvertierung braucht dann sehr lange (es wird immer das gesamte Items-Verzeichnis konvertiert) und man verändert direkt die Dateien auf dem sh.py-Server. Das könnte dazu führen, dass man sh.py nicht einfach so restarten kann, weil man gerade irgendwelche Items inkonsistent geändert hat.

Mein Vorschlag (so arbeite ich und die Anleitung bezieht sich auf die im folgenden beschriebene Projekt- bzw. Verzeichnisstruktur):

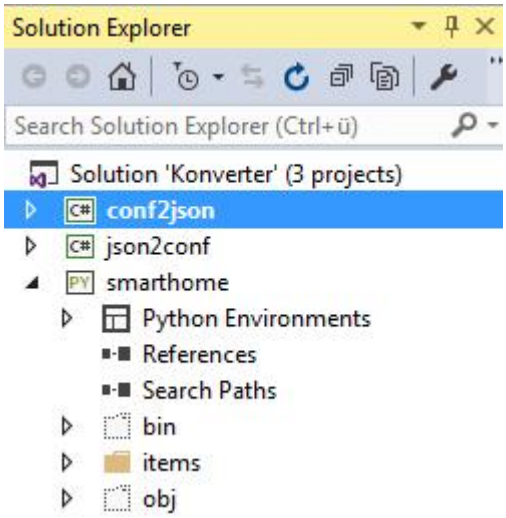
Kopiert ihr den kompletten Inhalt vom sh.py-items-Verzeichnis (also alle .conf-Files) in das items-Vezeichnis in dem Installationsverzeichnis (C:\Users \<username>\Source\Repos\smarthome-json-2-conf\smarthome\items)

Um dann einfach zwischen dem sh.py-items-Verzeichnis und dem Konverter-items-Verzeichnis zu synchronisieren, benutze ich FreeFileSync, zu finden unter https://www.freefilesync.org/

Auf diese Weise kann man alle Änderungen + Konvertierung lokal machen, und das iterativ so lange, bis der Konverter keine Fehler mehr meldet. Anschließend kann man das Ergebnis (die .conf-Files) mit FreeFileSync auf den sh.py server deployen und dann dort sh.py neu durchstarten. Mit FreeFileSync kann man sich auch einfach ein Batch-File gererieren lassen, so dass die gesamte Synchronisierung dann auf ein Doppelclick passieren kann. Wenn man es noch komfortabler machen will, integriert man den Aufruf auch noch in Visual Studio, aber das ist nicht part dieser Anleitung.

Egal für welches Verzeichnis mal sich entscheidet, in dem das smarthome-Verzeichnis samt aller Unterverzeichnisse liegt, man muss dem Konverter dieses Verzeichnis bekannt geben. Dies geht folgendermaßen:

Im Verzeichnis C:\Users\<username>\Source\Repos\smarthome-json-2-conf (im folgenden Konverter-Verzeichnis genammt) gibt es eine Konverter.sln, die wird mit doppelclick gestartet. Wenn alles geöffnet ist, ist rechts ein Fenster mit dem Namen "Soulution Explorer". Dort sollte "conf2json" fett hervorgehoben sein



(Falls nicht: Mit rechter Maustaste klicken und "Set as startup project" auswählen).

Diesen Knoten jetzt aufklappen und einen Doppelklick auf "Properties". In dem sich öffnenden Fenster auf der linken Seite "Debug" clicken. Jetzt muss man in dem großen Eingabefeld unter "Start Options", "command line arguments" den Pfad zum items-Verzeichnis angeben. Falls man die hier vorgeschlagene Projektstruktur verwendet, ist der Pfad

"..\..\..\smarthome\items\"

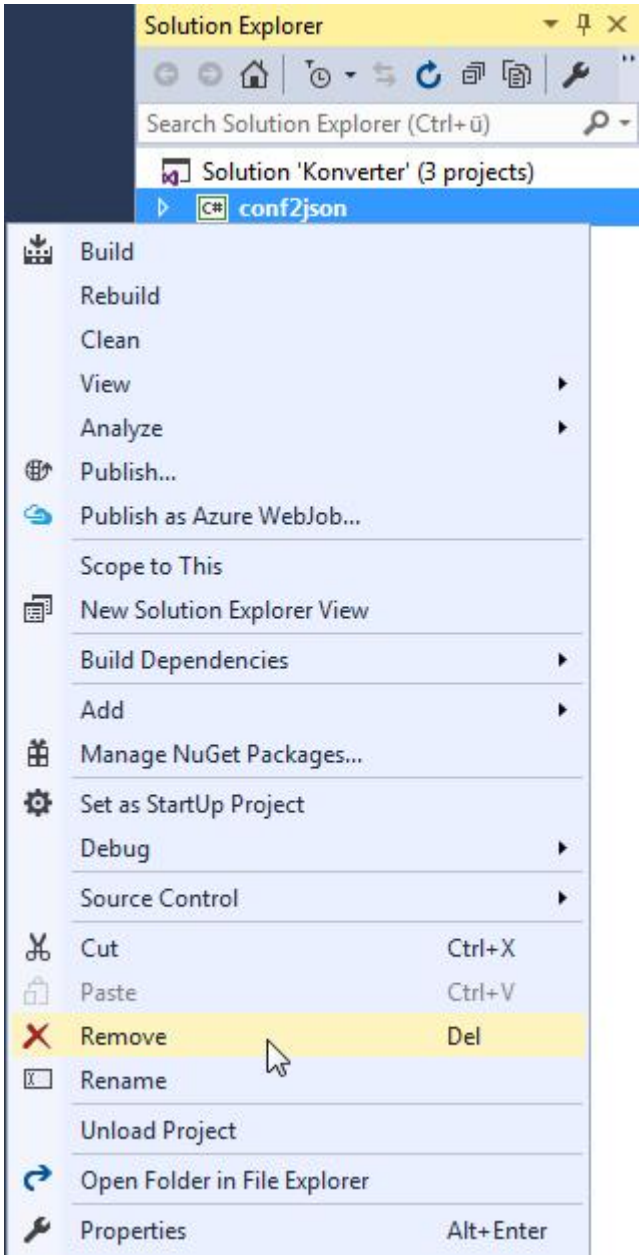
Das gleiche muss man mit json2conf->Properties, Debug, Start Options, command line options machen.

Konvertierung von .conf-Files nach json (EINMALIGER Schritt):

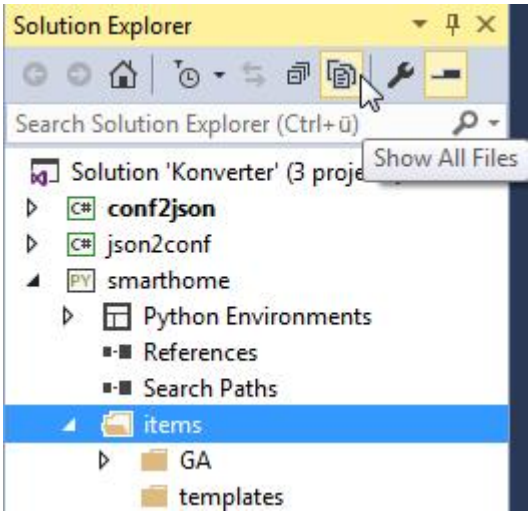
Nach einem check, ob conf2json wirklich fett hervorgehoben ist, bitte F5 drücken. Nun werden alle .conf-Files in .json-Files gewandelt. Dabei werden alle Item-Namen so umbenannt, dass sie mit einem Grossbuchstaben beginnen. Dieser Schritt ist EINMALIG, NICHT UMKEHRBAR und erstmal OHNE NEBENEFFEKTE, da nur neue Files erzeugt werden.

Spätestens jetzt solltet ihr eine Kopie des gesamten items-Verzeichnisses machen, damit ein Backup da ist.

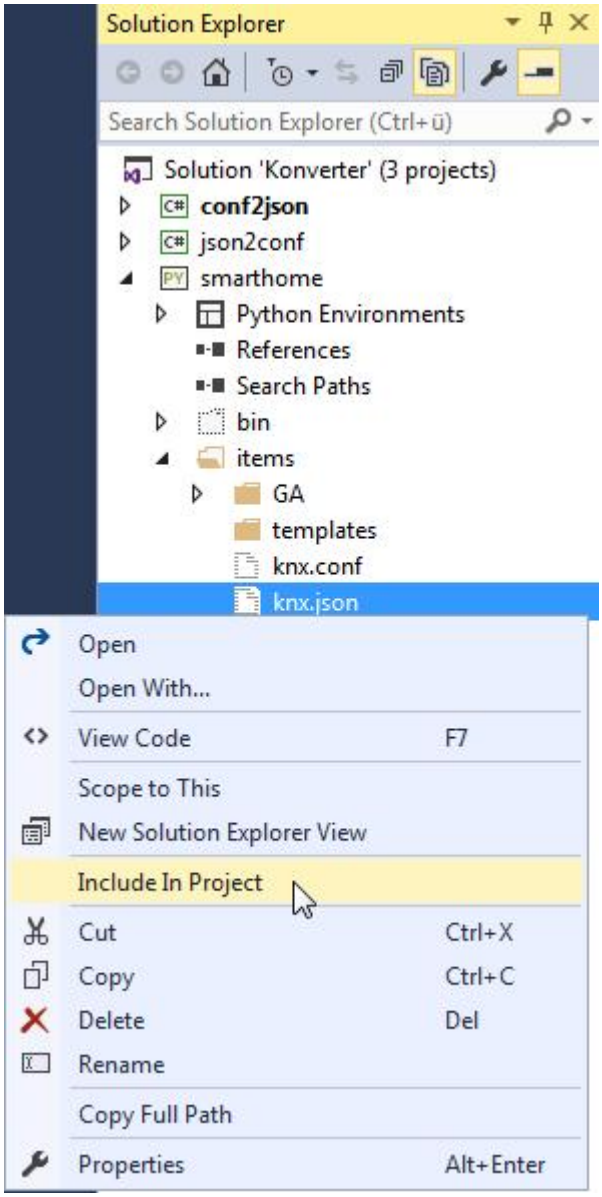
Jetzt mit rechter Maustaste aus "conf2json" clicken und "Remove" auswählen. Dieses Projekt wird aus der Solution entfernt (verbleibt aber auf Platte), conf2json wird nicht mehr gebraucht.



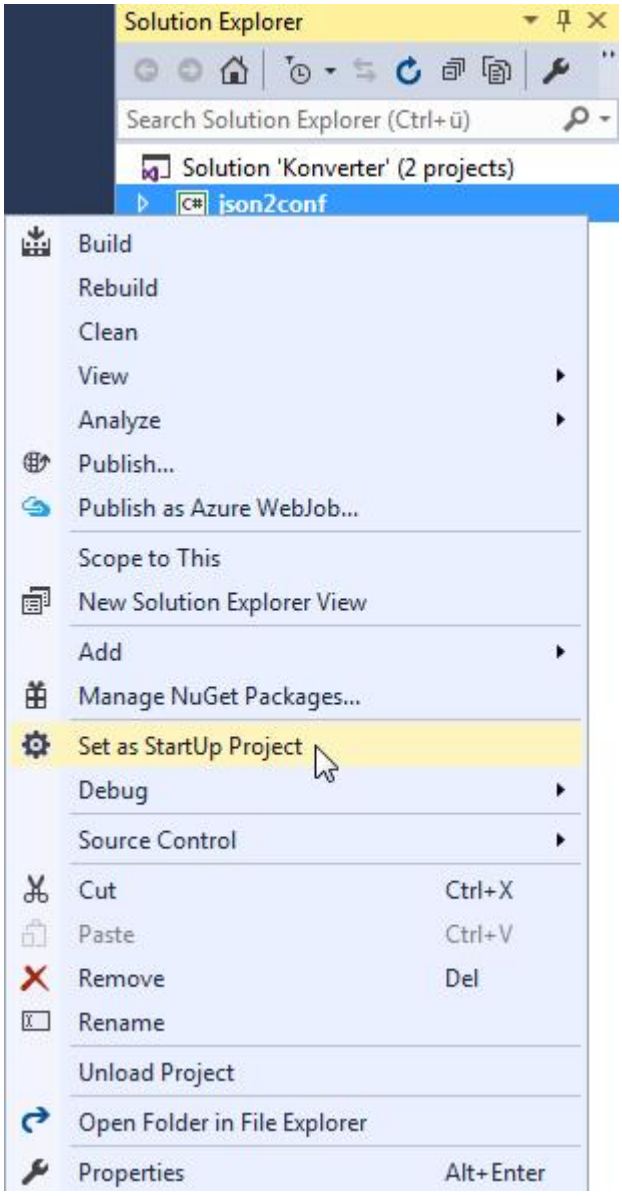
Nun im "Solution Explorer" das Projekt "smarthome" aufklappen und dann auch den Folder "items" aufklappen. Den Folder "items" auch mit der linken Maustaste anklicken, er muss markiert (blau hervorgehoben) sein. Jetzt in der Toolbar vom "Solution Explorer" das Icon "Show all Files" anklicken.



Jetzt sollte man unter "items" alle *.conf- und *.json-Files sehen. Bitte jedes *.json-File mit der rechten Maustaste anklicken und "Include in Project" anklicken.

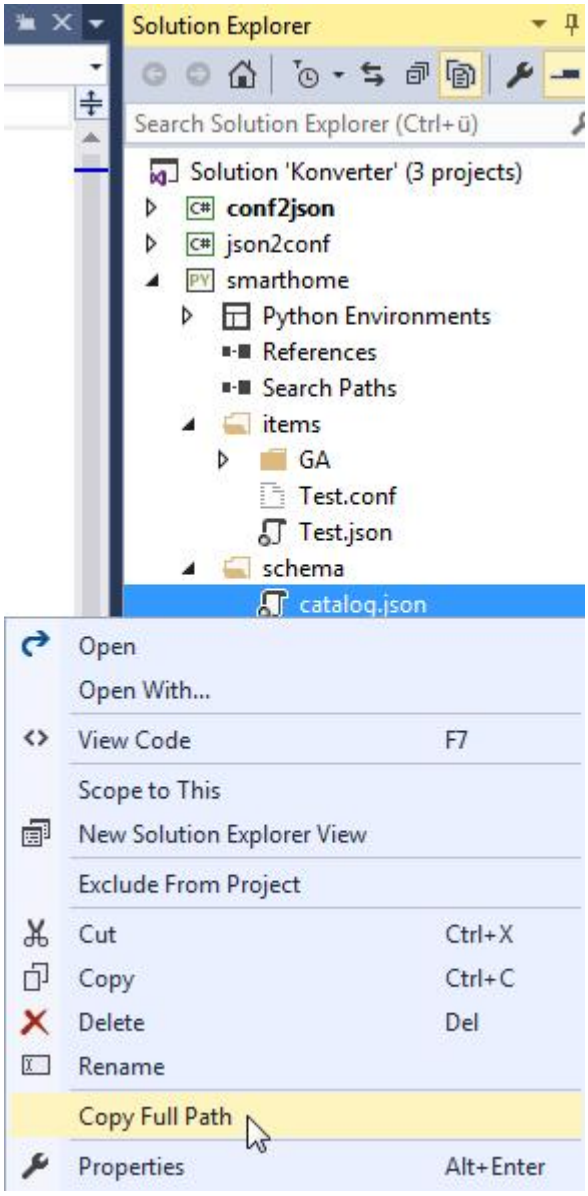


Jetzt mit rechter Maustaste auf "json2conf" clicken und "Set as startup project" auswählen.

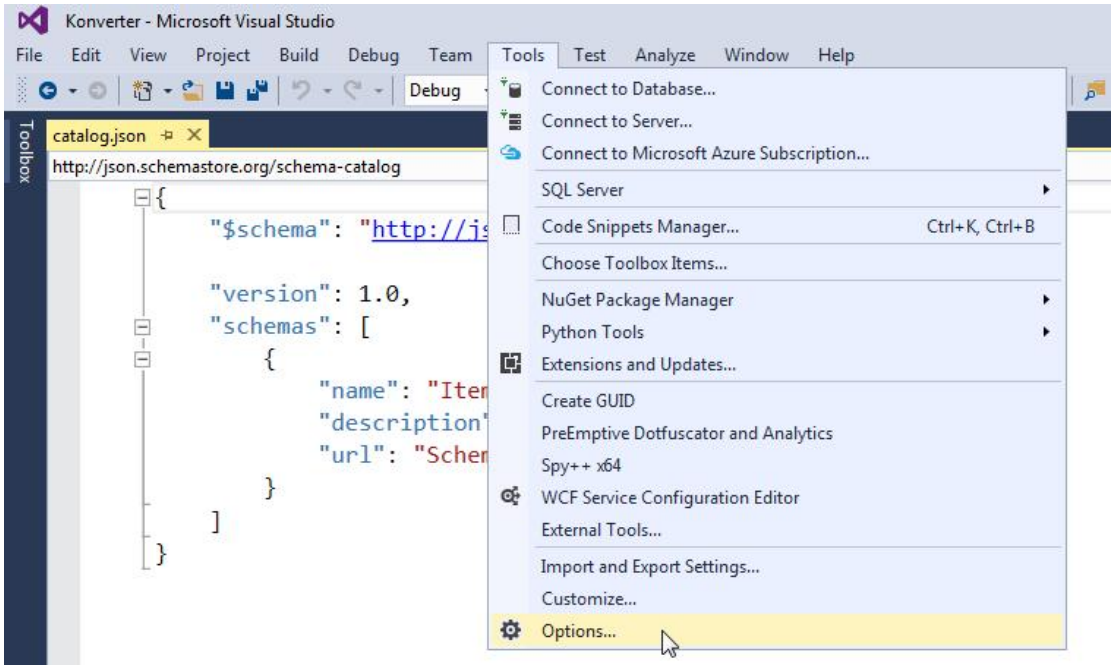


Jetzt kommt leider eine etwas unschöne Sache: Derzeit unterstützt der json-Editor Schema-Files, aber diese müssen von einer URL kommen. Bei Files von der Platte funktioniert das noch nicht so gut. Es gibt aber eine Dropdown, mit der man Schemata auswählen kann, und in diese schreiben wir das von uns verwendete Schema rein.

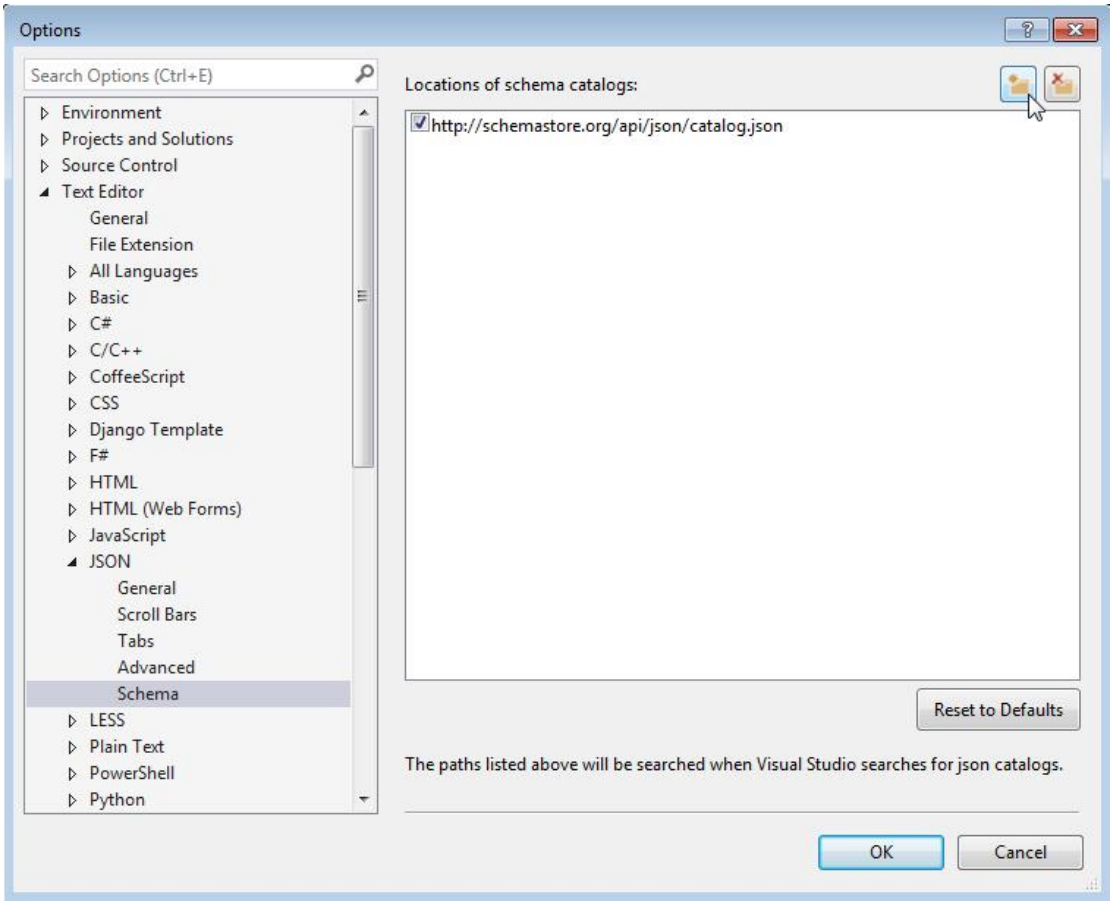
Dazu müssen wir im Solution Explorer mit der rechten Maustaste auf das File catalog.json in smarthome/schema clicken und "Copy full path" auswählen. Jetzt haben wir den Pfad der Datei im Clipboard.



Nun müssen wir im Hauptmenu Tools->Options... auswählen

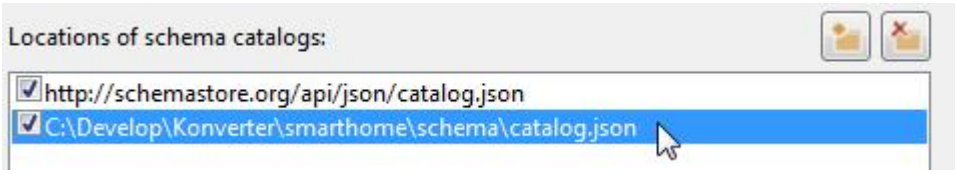


In dem darauffolgenden Dialog bitte im linken Baum "Text Editor->JSON-> Schema" auswählen und dann rechts oben auf "Add Folder" clicken

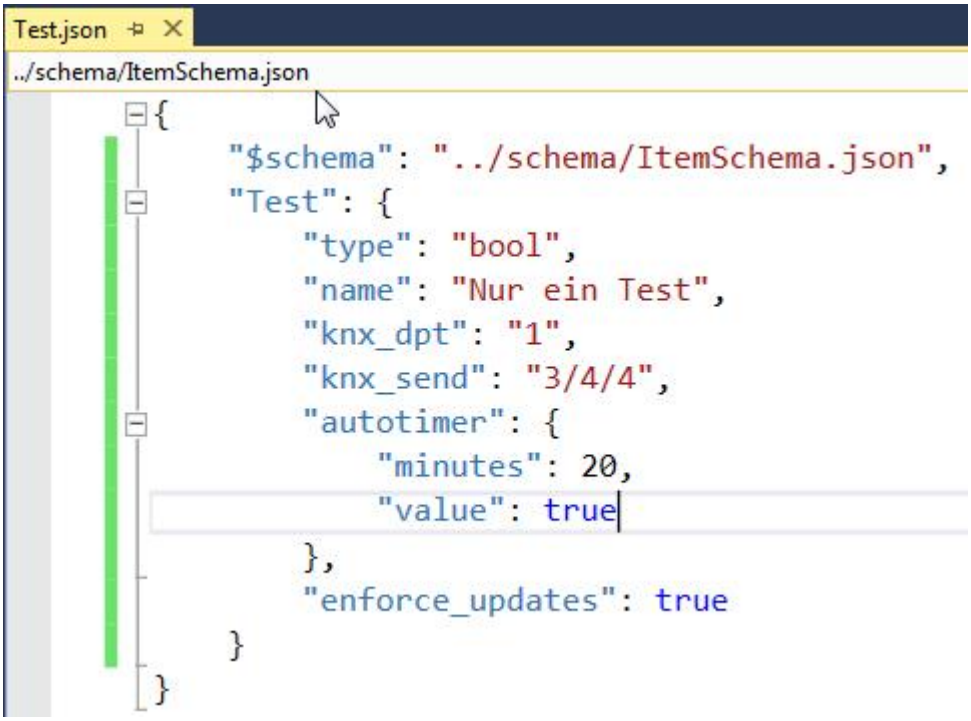


In der dann erscheinenden Textbox Strg-V (Paste) drücken und den Clipboardinhalt einfügen. Unbedingt <ENTER> drücken, bevor man auf OK

clickt (Bei euch sollte der Pfad natürlich anders sein).



Das wars! Um zu überprüfen, ob das alles geklappt hat, öffne bitte jetzt eines der vom conf2json erzeugten .json-Files (im Folder smarthome/items) durch doppelclick. Das File erscheint im Editor und oben ist eine Dropdown, in der `"../schema/ItemSchema.json"` steht.



Jetzt bitte SaveAll clicken (Ctrl+Shift+S). Nun Visual Studio beenden - das initiale Setup ist fertig.